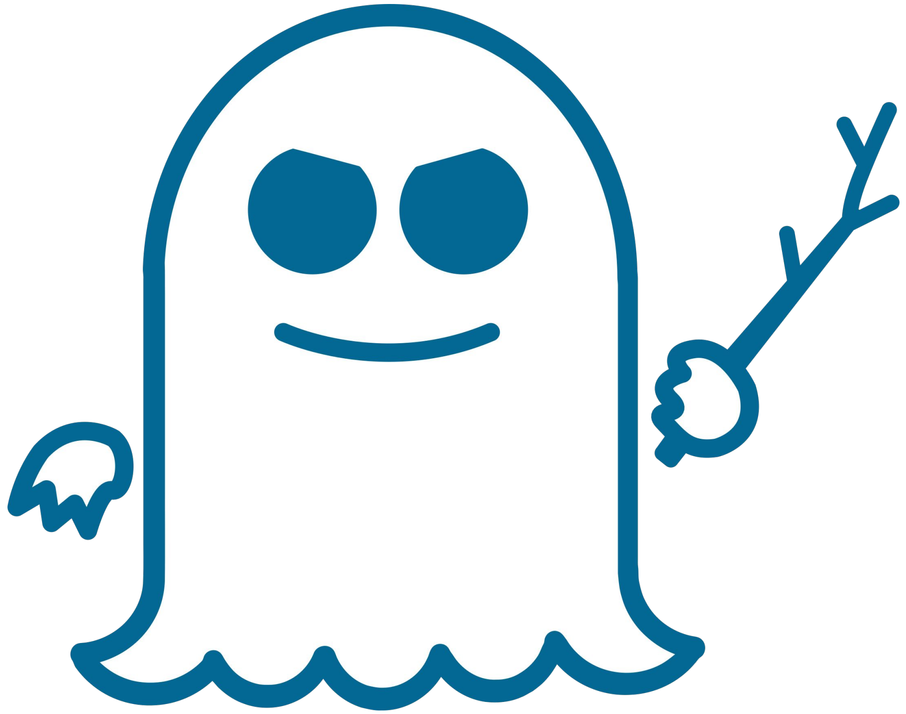
SpecFuzz

Bringing Spectre-type vulnerabilities to the surface

Oleksii Oleksenko

Motivation

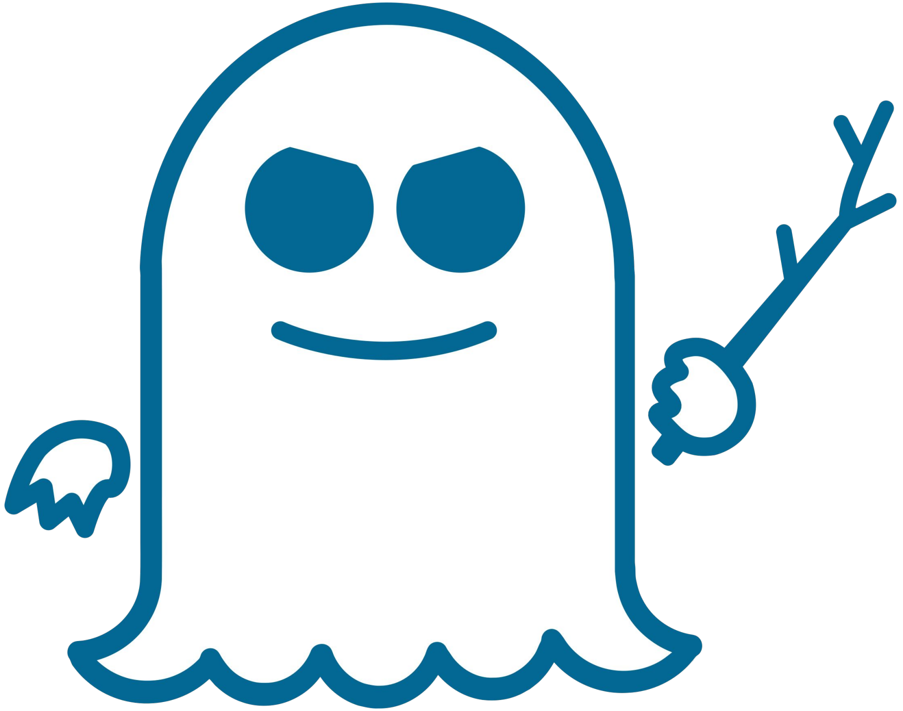
BOO!



Not scary
at all?

Cute?

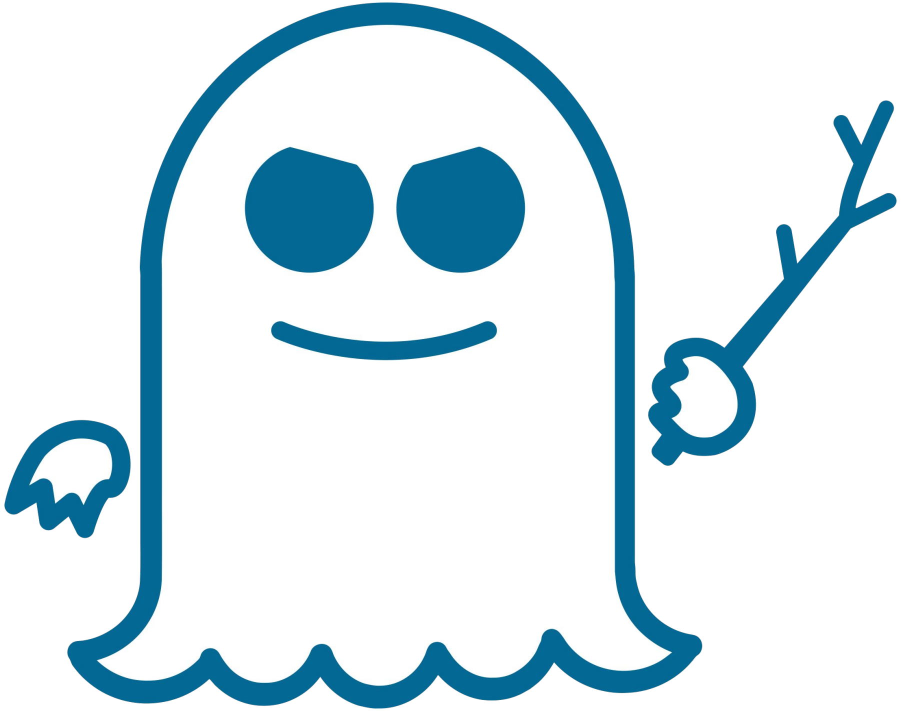
Cuddly?



Not scary **at all**?

Cute?

Cuddly?



Buffer overflow

```
y = array[x];
```

x can be
larger than
the `array` size

Bounds check

```
if (x >= 0 && x < size) {
    y = array[x];
}
```

Fixed!

I SHALL BYPASS!

x = 1

```
// size = 10
if (x >= 0 && x < size) {  True
    y = array[x];
}
```

x = 3

```
// size = 10
if (x >= 0 && x < size) {  True
    y = array[x];
}
```

x = 2

```
// size = 10
if (x >= 0 && x < size) {
    y = array[x];
}
```

True

x = 1 Gazillion

```
// size = 10
if (x >= 0 && x < size) {  ???
    y = array[x];
}
```

BOUNDS CHECKS
BRANCH PREDICTOR

What will happen to us?
BOUNDS CHECKS
BRANCH PREDICTOR

BOUNDS CHECKS
BRANCH PREDICTOR
Hm... Let me see your history

BOUNDS CHECKS
BRANCH PREDICTOR
You passed once, then again, and again...

BOUNDS CHECKS
BRANCH PREDICTOR
You will pass!

Bounds check bypass

Predict true

Execute speculatively

```
if (x >= 0 && x < size) {
    y = array[x];
}
```

Bounds check bypass

Predict true

Execute speculatively

```
if (x >= 0 && x < size) {
    y = array[x];
}
```

Speculative execution:

- Not visible to software
- Leaves **detectable** traces in hardware

Isn't it a CPU bug?

CPU Model and Stepping	V1, Spectre	V2, Spectre	V3, Meltdown	V3a	V4	L1TF, Foreshadow	MFBDS, RIDL
Intel64 Family 6 Model 142 Stepping 11	Software	MCU + Software	Hardware	MCU	MCU + Software	Hardware	Hardware
Intel64 Family 6 Model 142 Stepping 12	Software	Hardware + Software	Hardware	MCU	Hardware + Software	Hardware	Hardware
Intel64 Family 6 Model 158 Stepping 11	Software	MCU + Software	Software	MCU	MCU + Software	MCU + Software	MCU + Software
Intel64 Family 6 Model 158 Stepping 12	Software	MCU + Software	Hardware	MCU	MCU + Software	Hardware	Hardware
Intel64 Family 6 Model 158 Stepping 13	Software	Hardware + Software	Hardware	MCU	Hardware + Software	Hardware	Hardware

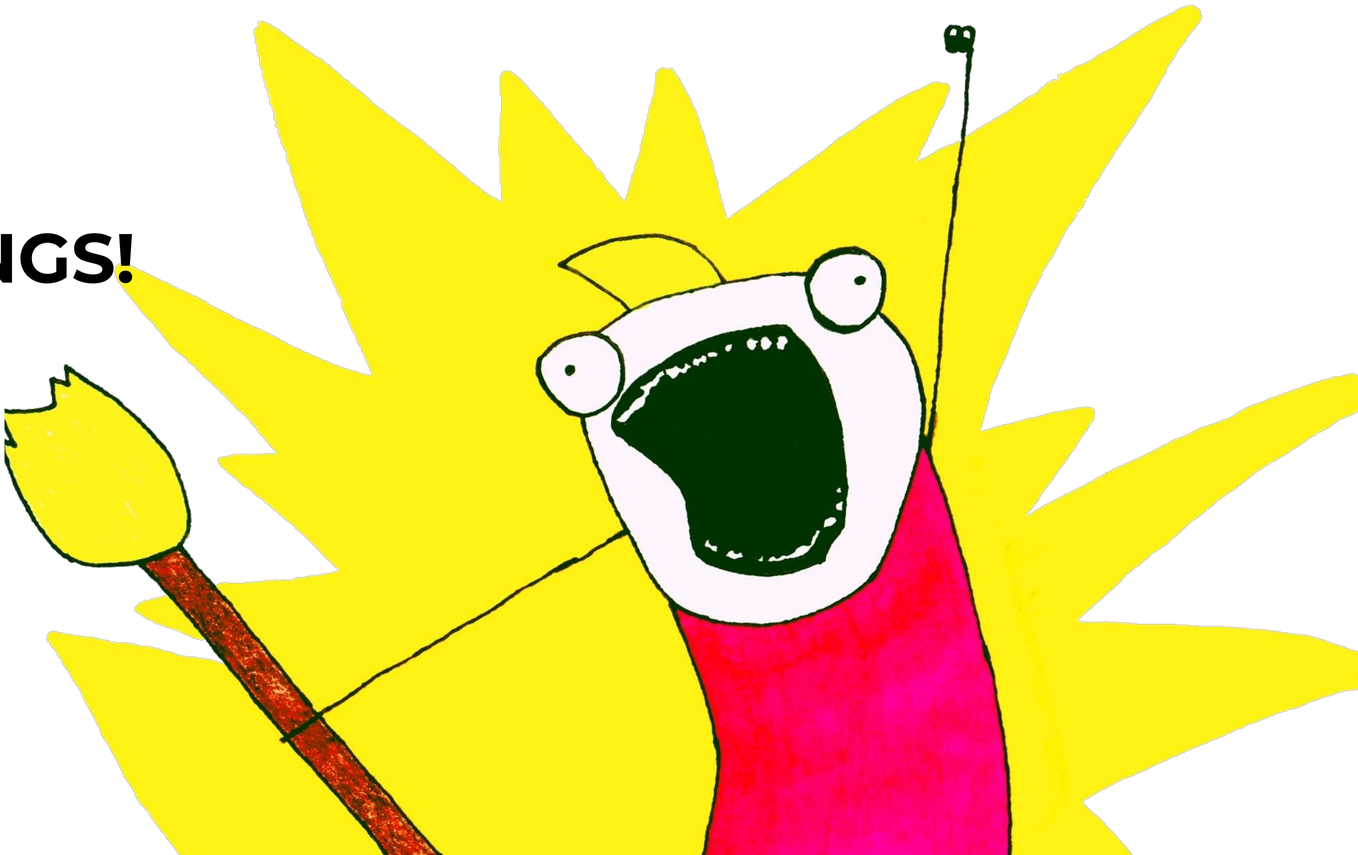
CPU Model and Stepping	V1, Spectre	V2, Spectre	V3, Meltdown	V3a	V4	L1TF, Foreshadow	MFBDS, RIDL
Intel64 Family 6 Model 142 Stepping 11	Software	MCU + Software	Hardware	MCU	MCU + Software	Hardware	Hardware
Intel64 Family 6 Model 142 Stepping 12	Software	Hardware + Software	Hardware	MCU	Hardware + Software	Hardware	Hardware
Intel64 Family 6 Model 158 Stepping 11	Software	MCU + Software	Software	MCU	MCU + Software	MCU + Software	MCU + Software
Intel64 Family 6 Model 158 Stepping 12	Software	MCU + Software	Hardware	MCU	MCU + Software	Hardware	Hardware
Intel64 Family 6 Model 158 Stepping 13	Software	Hardware + Software	Hardware	MCU	Hardware + Software	Hardware	Hardware

SPECTRE V1 IS A FLAW IN OUT PRODUCTS
PRETTY MUCH ANY CPU VENDOR
BUT THAT'S NONE OF OUR BUSINESS

What can I do?

SERIALIZE

ALL THE THINGS!



```
if (x >= 0 && x < size) {
   __mm_lfence();   // stops speculation
    y = array[x];
}
```

```
if (x >= 0 && x < size) {
  __mm_lfence(); // stops speculation
  y = array[x];
}
```

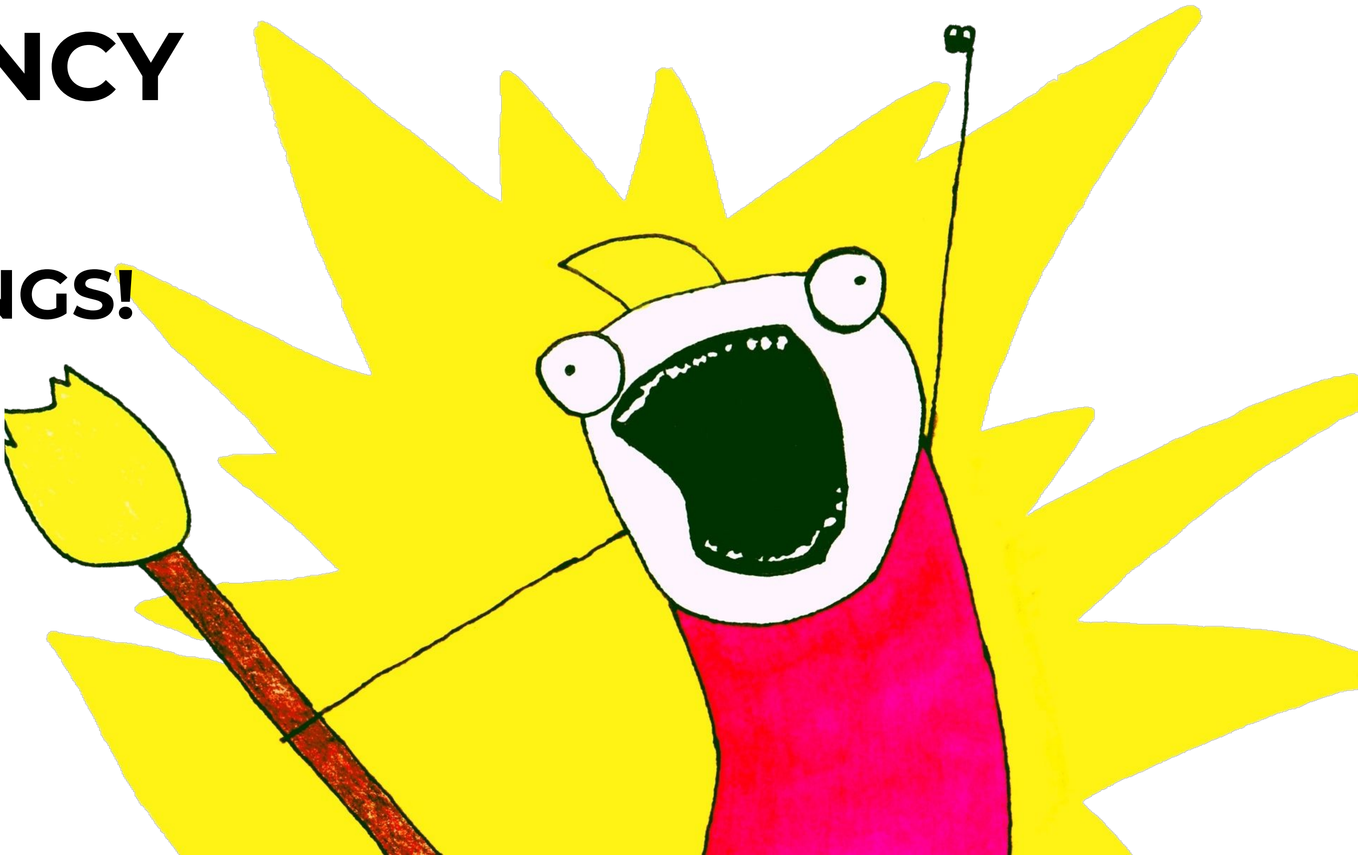
400% SLOWDOWN

ADD

DATA DEPENDENCY

TO

ALL THE THINGS!



```
if (x < size) {
   x = (x < size) ? x : 0;
   y = array[x];
}
```

```
if (x < size) {
    x = (x < size) ? x : 0;
    y = array[x];
}
```

50% SLOWDOWN

We need more precision!

How do we find Spectre?

How do we find other bugs?

Fuzzing!

```
x = generate_randomized_int();

if (x >= 0 && x < size) {

   y = array[x];
}
```

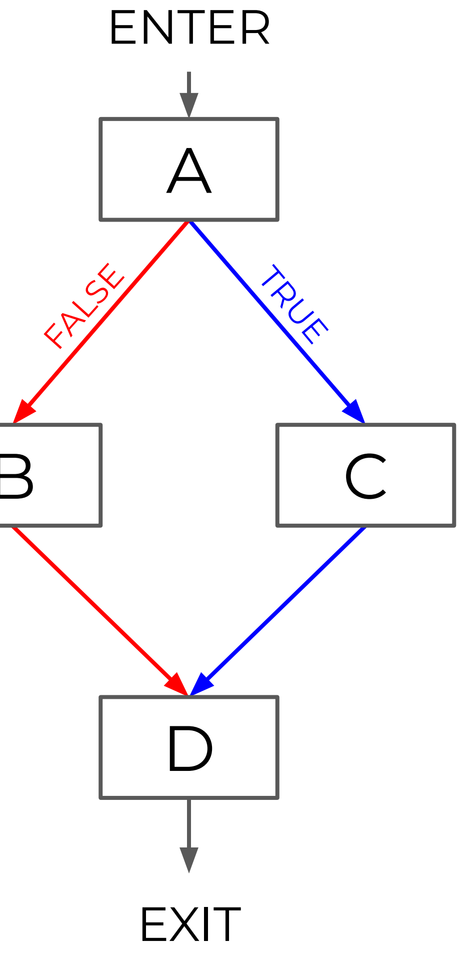
```
x = generate_randomized_int();

if (x >= 0 && x < size) {
   __asan_check_if_valid(array + x);
    y = array[x];
}
```

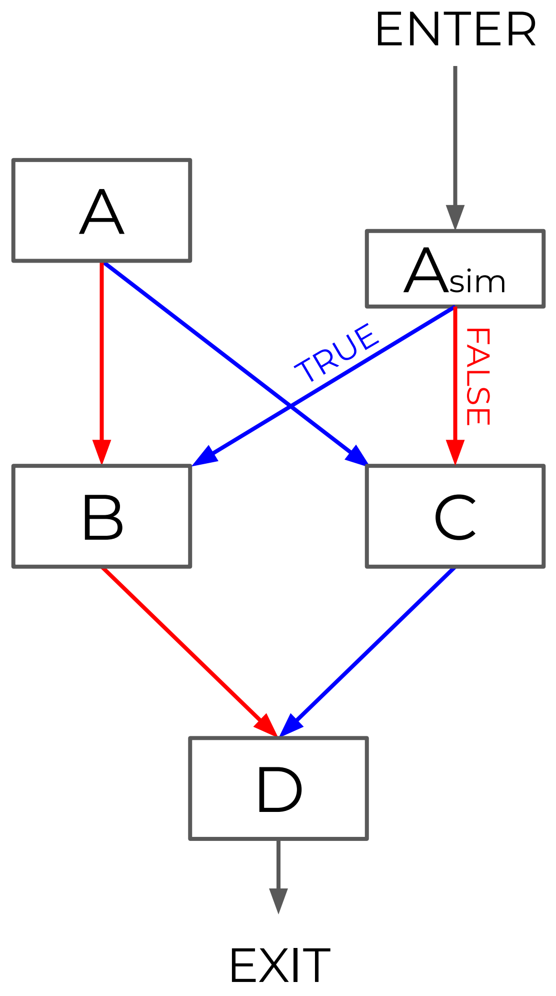
```
x = generate_randomized_int();

if (x >= 0 && x < size) {
    __asan_check_if_valid(array + x);
    y = array[x];
}
```

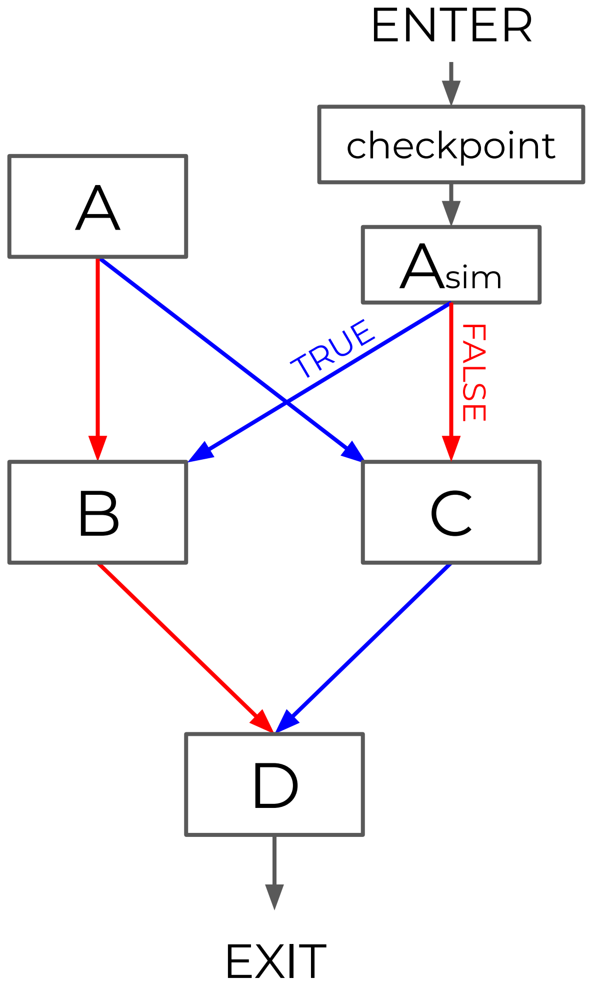
Always valid!

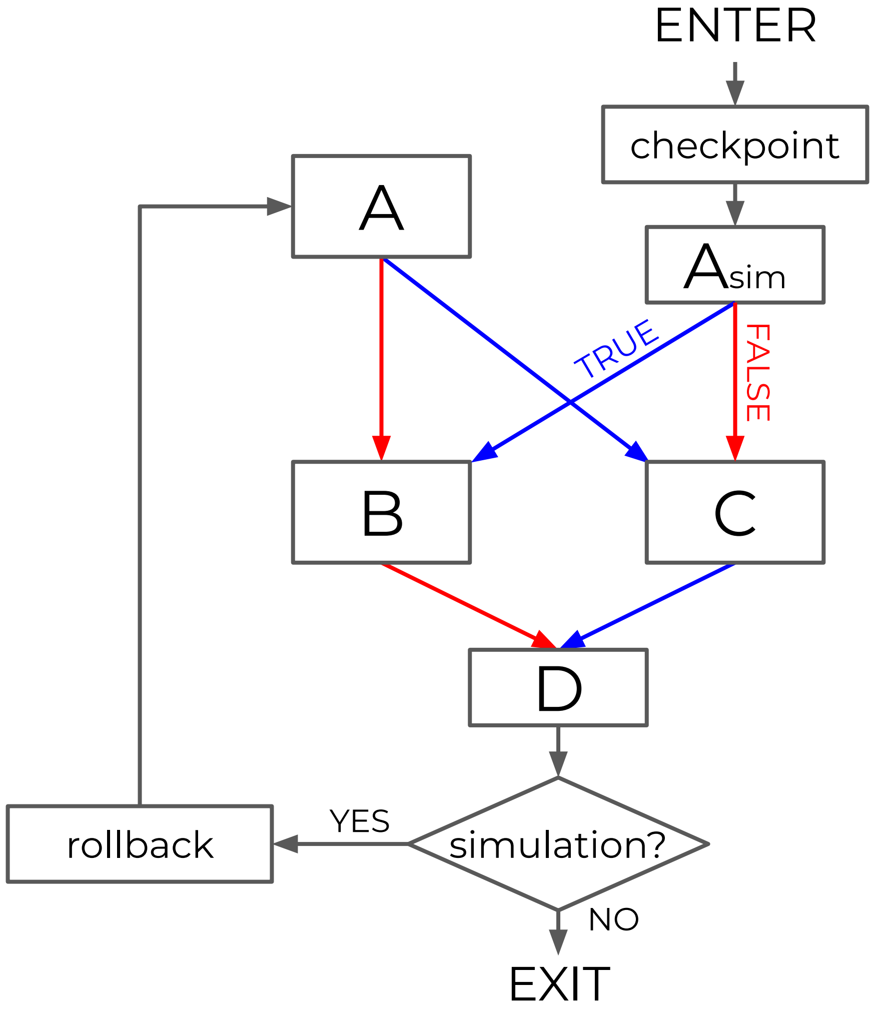
How do we make speculative execution visible?

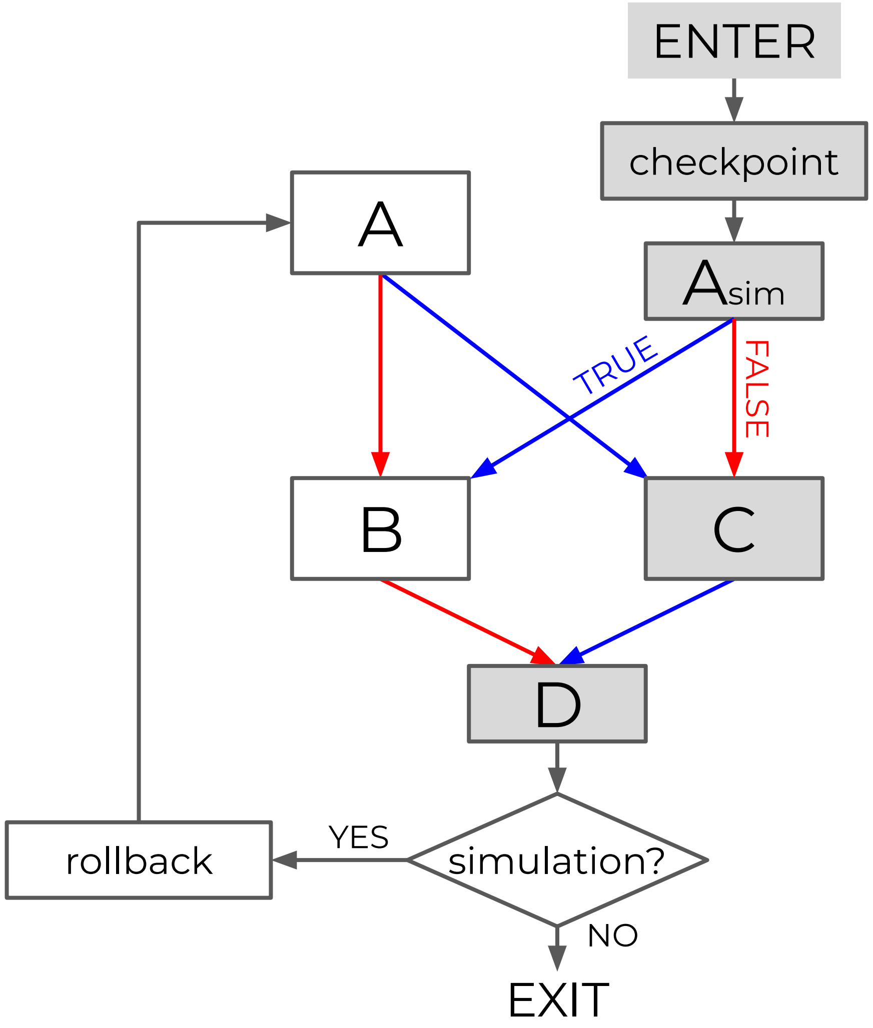
Let's simulate it!

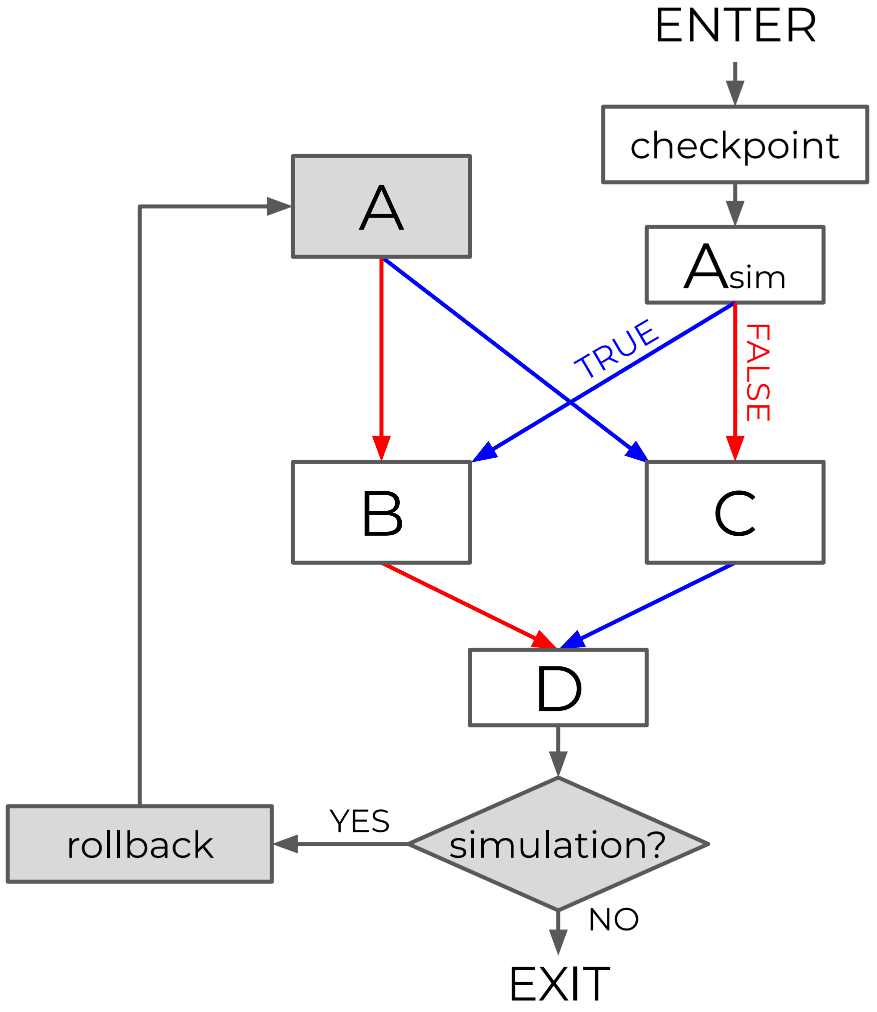

ENTER
A
FALSE
TRUE
B
C
D
EXIT

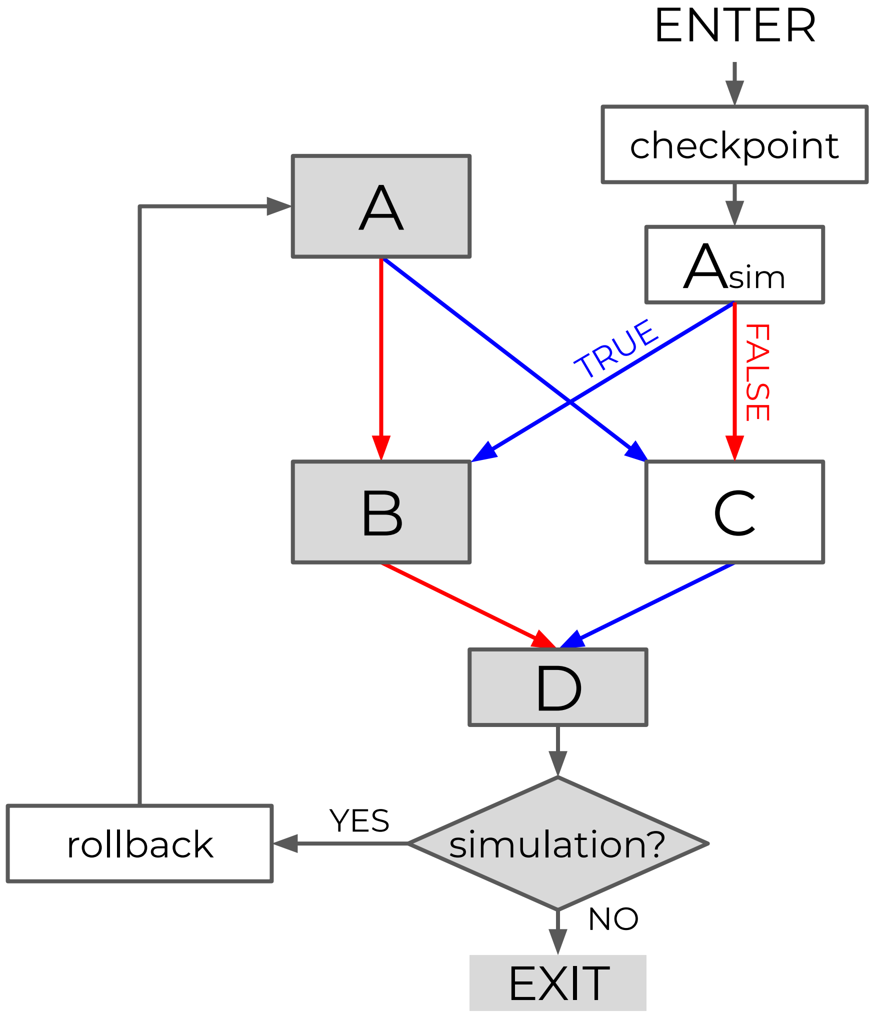

ENTER
A
Asim
TRUE
FALSE
B
C
D
EXIT


ENTER
checkpoint
A
Asim
TRUE
FALSE
B
C
D
EXIT


ENTER
checkpoint
A
A sim
TRUE
FALSE
B
C
D
simulation?
YES
rollback
NO
EXIT


ENTER
checkpoint
A
A_{sim}
TRUE
FALSE
B
C
D
simulation?
YES
rollback
NO
EXIT


ENTER
checkpoint
A
A_{sim}
TRUE
FALSE
B
C
D
simulation?
YES
NO
rollback
EXIT


ENTER
checkpoint
A
Asim
TRUE
FALSE
B
C
D
simulation?
YES
rollback
NO
EXIT

Example

```
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
        CMP %rdi, size
.if:    JL  .else
        MOV array(%rdi), %eax
        AND %al, result
.else:  RET
```

```c
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
        CMP %rdi, size
.if:    JL  .else
        MOV array(%rdi), %eax
        AND %al, result
.else:  RET
```

```
<victim_function>:
        CMP  %rdi, size


.if:    JL   .else



        MOV  (%rdi), %eax
        AND  %al, result

.else:  RET
```

```
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
        CMP %rdi, size
.if:    JL  .else
        MOV array(%rdi), %eax
        AND %al, result
.else:  RET
```

```
<victim_function>:
        CMP  %rdi, size
        CALL specfuzz_chkp


.if:    JL   .else



        MOV  (%rdi), %eax
        AND  %al, result

.else:  RET
```

Checkpoint + mispredict

```c
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
        CMP %rdi, size
.if:    JL  .else
        MOV array(%rdi), %eax
        AND %al, result
.else:  RET
```

```
<victim_function>:
        CMP  %rdi, size
        CALL specfuzz_chkp
        JGE  .else
        JMP  .skip
.if:    JL   .else
.skip:


        MOV  (%rdi), %eax
        AND  %al, result

.else:  RET
```

Checkpoint + mispredict

```c
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
        CMP %rdi, size
.if:    JL  .else
        MOV array(%rdi), %eax
        AND %al, result
.else:  RET
```

```
<victim_function>:
        CMP  %rdi, size
        CALL specfuzz_chkp          }
        JGE  .else                  }  Checkpoint +
        JMP  .skip                  }  mispredict
.if:    JL   .else
.skip:


        MOV  (%rdi), %eax
        AND  %al, result
        CALL specfuzz_maybe_rlbk    }  Rollback
.else:  RET
```

```c
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
         CMP %rdi, size
.if:     JL  .else
         MOV array(%rdi), %eax
         AND %al, result
.else:   RET
```

```
<victim_function>:
         CMP  %rdi, size
         CALL specfuzz_chkp
         JGE  .else
         JMP  .skip
.if:     JL   .else
.skip:   SUB  $0x2, instruction_counter

         MOV  (%rdi), %eax
         AND  %al, result
         CALL specfuzz_maybe_rlbk
.else:   RET
```

Checkpoint + mispredict

Rollback if counter > 250

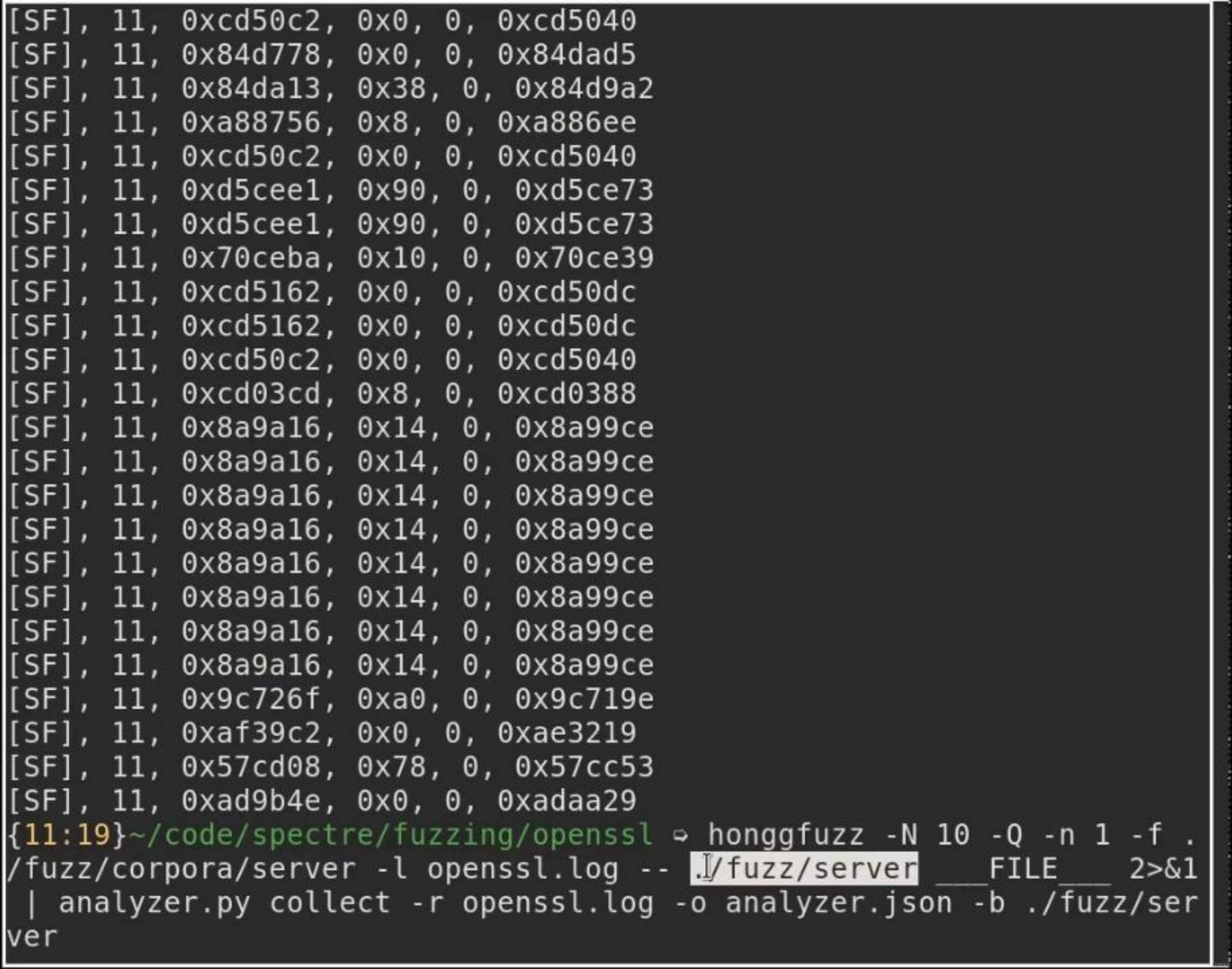
```
void victim_function(size_t x) {
    if (x < size) {
        result &= array[x];
    }
}
```

```
<victim_function>:
        CMP %rdi, size
.if:    JL  .else
        MOV array(%rdi), %eax
        AND %al, result
.else:  RET
```

```
<victim_function>:
        CMP  %rdi, size
        CALL specfuzz_chkp
        JGE  .else                           Checkpoint +
        JMP  .skip                           mispredict
.if:    JL   .else
.skip:  SUB  $0x2, instruction_counter
        LEA  array(%rdi), %rdi               Bounds check
        CALL __asan_load1
        MOV  (%rdi), %eax
        AND  %al, result
        CALL specfuzz_maybe_rlbk             Rollback if
.else:  RET                                  counter > 250
```

Demo

Fuzzing OpenSSL



```
[SF], 11, 0xcd50c2, 0x0, 0, 0xcd5040
[SF], 11, 0x84d778, 0x0, 0, 0x84dad5
[SF], 11, 0x84da13, 0x38, 0, 0x84d9a2
[SF], 11, 0xa88756, 0x8, 0, 0xa886ee
[SF], 11, 0xcd50c2, 0x0, 0, 0xcd5040
[SF], 11, 0xd5cee1, 0x90, 0, 0xd5ce73
[SF], 11, 0xd5cee1, 0x90, 0, 0xd5ce73
[SF], 11, 0x70ceba, 0x10, 0, 0x70ce39
[SF], 11, 0xcd5162, 0x0, 0, 0xcd50dc
[SF], 11, 0xcd5162, 0x0, 0, 0xcd50dc
[SF], 11, 0xcd50c2, 0x0, 0, 0xcd5040
[SF], 11, 0xcd03cd, 0x8, 0, 0xcd0388
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x8a9a16, 0x14, 0, 0x8a99ce
[SF], 11, 0x9c726f, 0xa0, 0, 0x9c719e
[SF], 11, 0xaf39c2, 0x0, 0, 0xae3219
[SF], 11, 0x57cd08, 0x78, 0, 0x57cc53
[SF], 11, 0xad9b4e, 0x0, 0, 0xadaa29
{11:19}~/code/spectre/fuzzing/openssl ➭ honggfuzz -N 10 -Q -n 1 -f .
/fuzz/corpora/server -l openssl.log -- ./fuzz/server ___FILE___ 2>&1
 | analyzer.py collect -r openssl.log -o analyzer.json -b ./fuzz/ser
ver
```

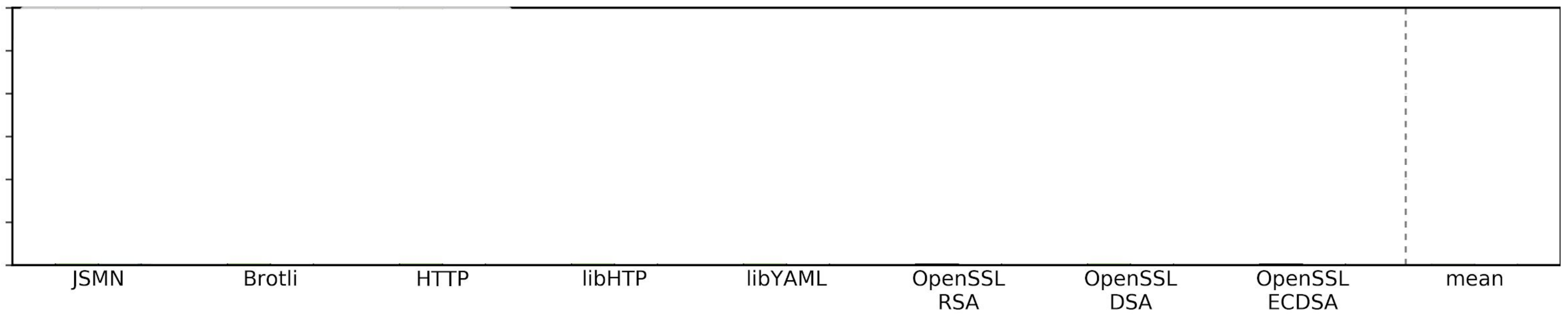
Now what?

Whitelist patching

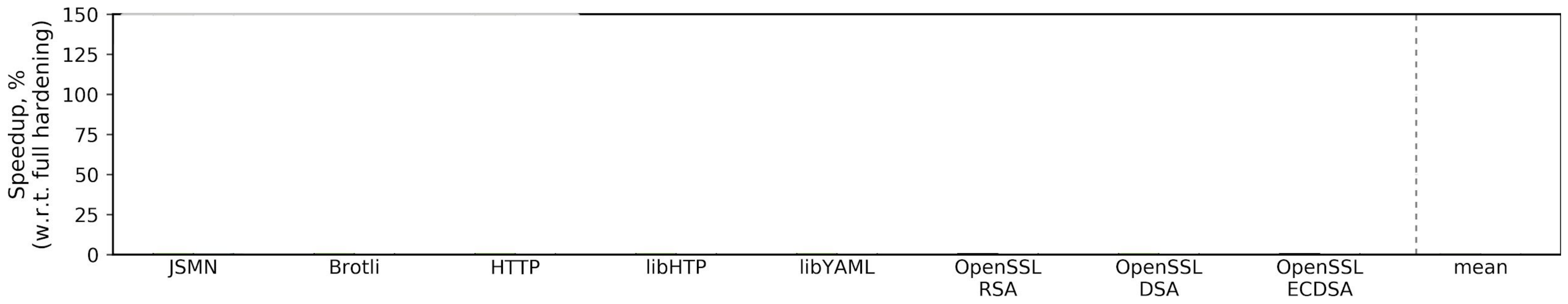
Instrument all branches except:

- Covered
- No vulnerabilities detected

Speedup

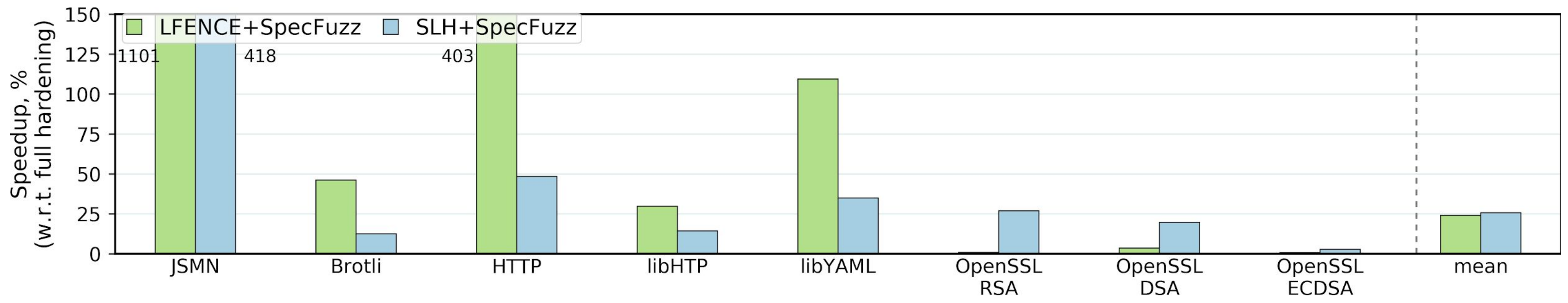

JSMN
Brotli
HTTP
libHTP
libYAML
OpenSSL RSA
OpenSSL DSA
OpenSSL ECDSA
mean

Speedup



Higher is better

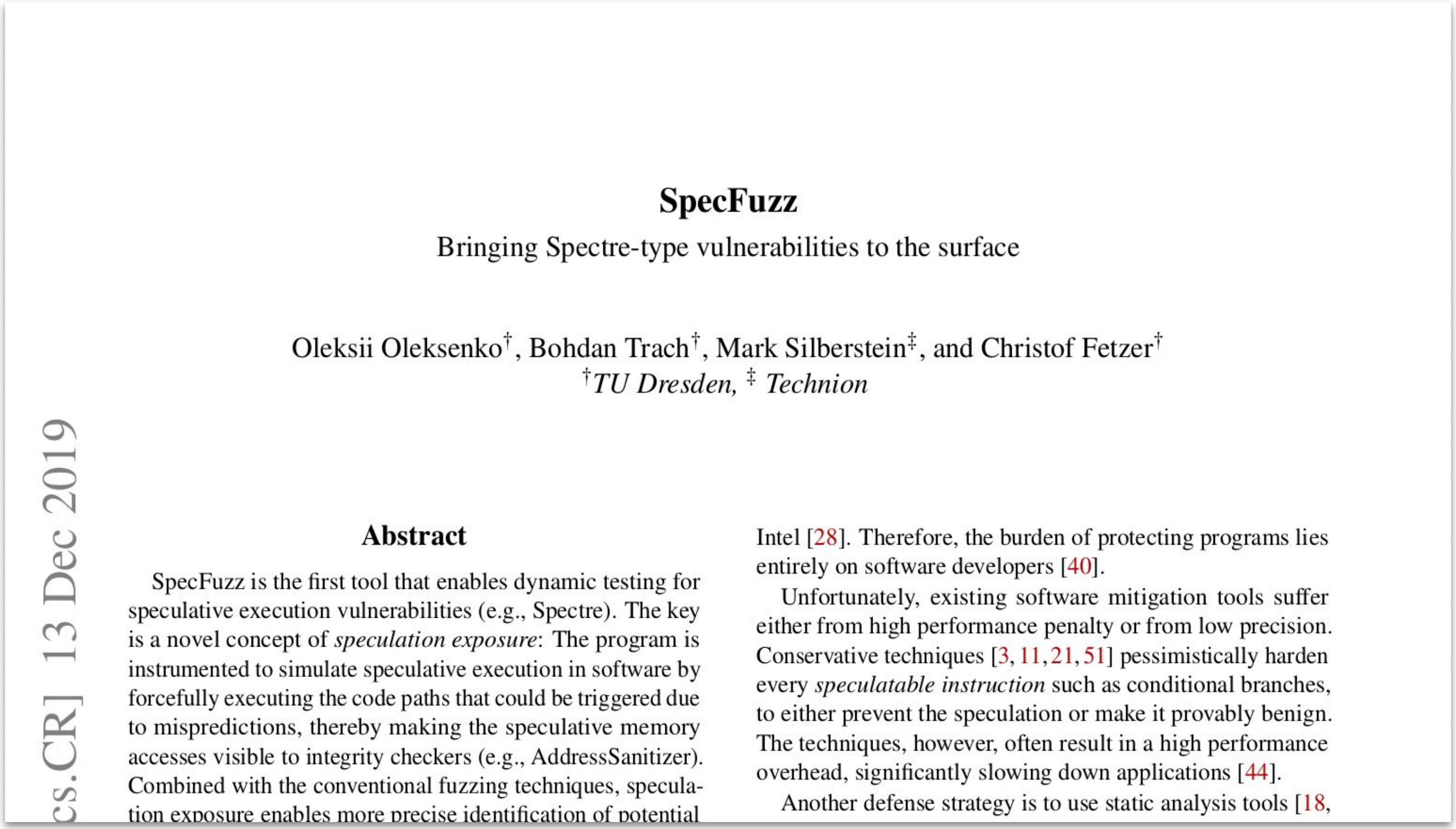
Speedup



Higher is better

Want more?

See our paper!



SpecFuzz

Bringing Spectre-type vulnerabilities to the surface

Oleksii Oleksenko[†], Bohdan Trach[†], Mark Silberstein[‡], and Christof Fetzer[†]
[†]*TU Dresden,* [‡] *Technion*

Abstract

SpecFuzz is the first tool that enables dynamic testing for speculative execution vulnerabilities (e.g., Spectre). The key is a novel concept of *speculation exposure*: The program is instrumented to simulate speculative execution in software by forcefully executing the code paths that could be triggered due to mispredictions, thereby making the speculative memory accesses visible to integrity checkers (e.g., AddressSanitizer). Combined with the conventional fuzzing techniques, speculation exposure enables more precise identification of potential

Intel [28]. Therefore, the burden of protecting programs lies entirely on software developers [40].

Unfortunately, existing software mitigation tools suffer either from high performance penalty or from low precision. Conservative techniques [3, 11, 21, 51] pessimistically harden every *speculatable instruction* such as conditional branches, to either prevent the speculation or make it provably benign. The techniques, however, often result in a high performance overhead, significantly slowing down applications [44].

Another defense strategy is to use static analysis tools [18, GitHub

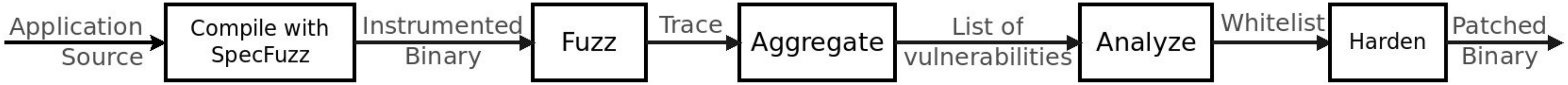
https://github.com/tudinfse/SpecFuzz

GitHub

https://github.com/tudinfse/SpecFuzz

Warning!
Academic Code

SpecFuzz

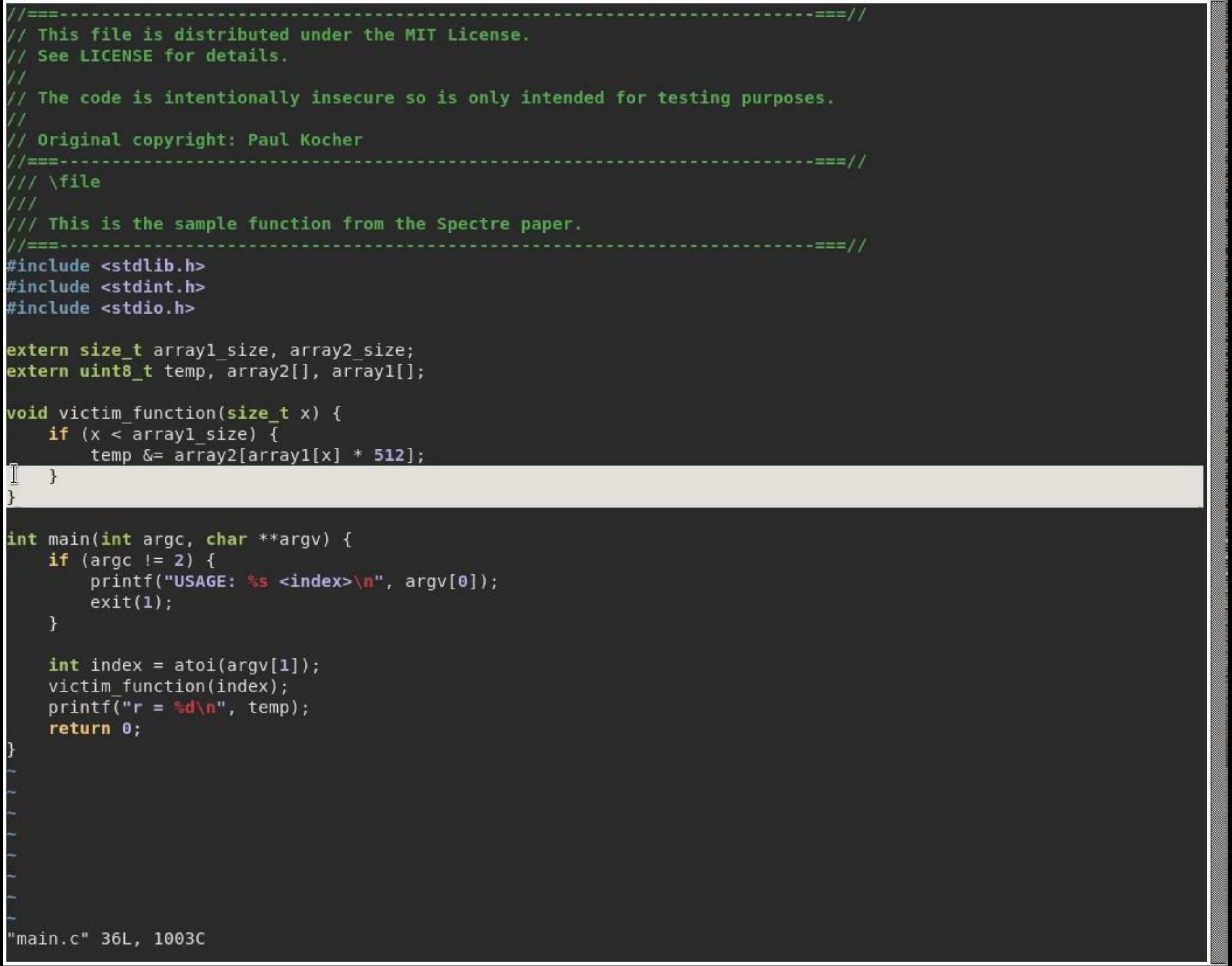


Questions?

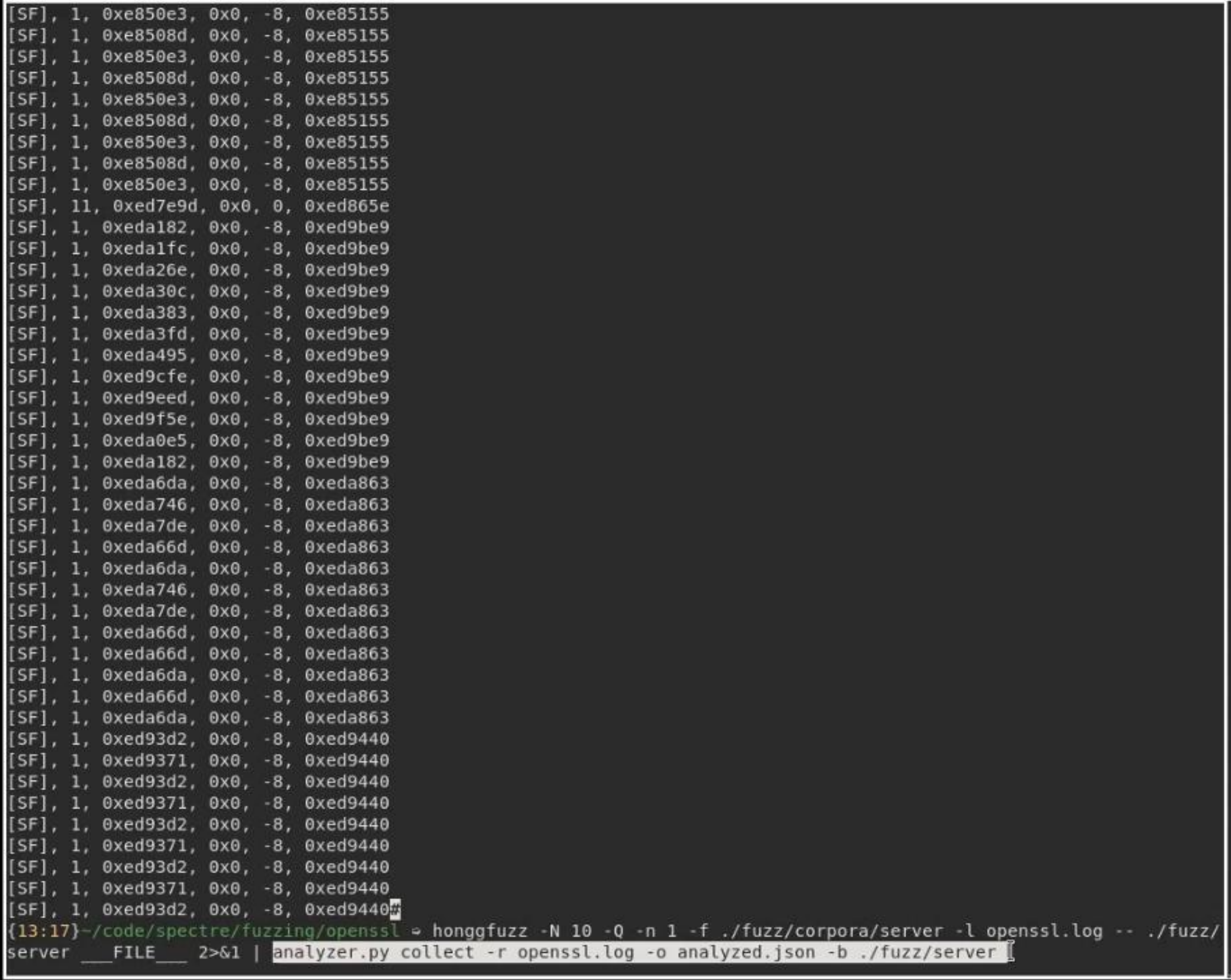
@oleksii_o

oleksii.oleksenko@tu-dresden.de

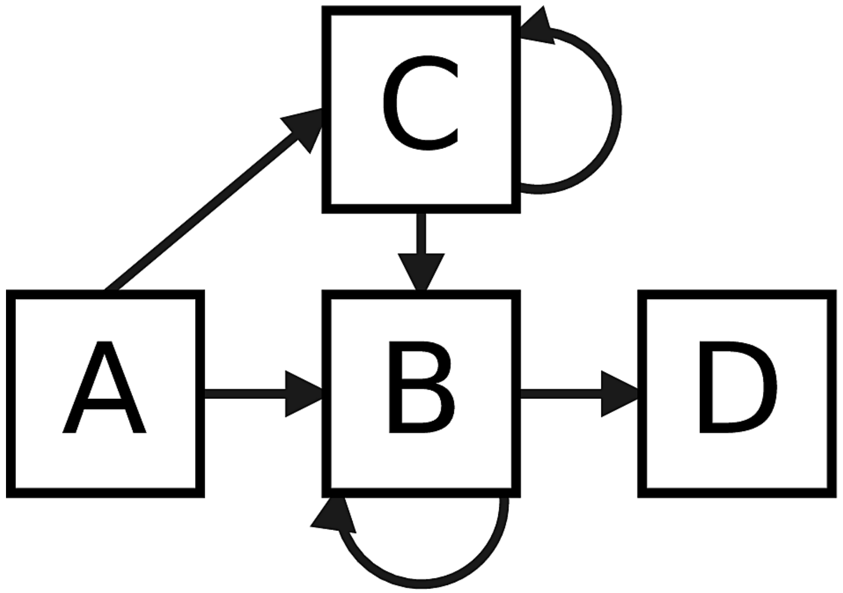
Backup



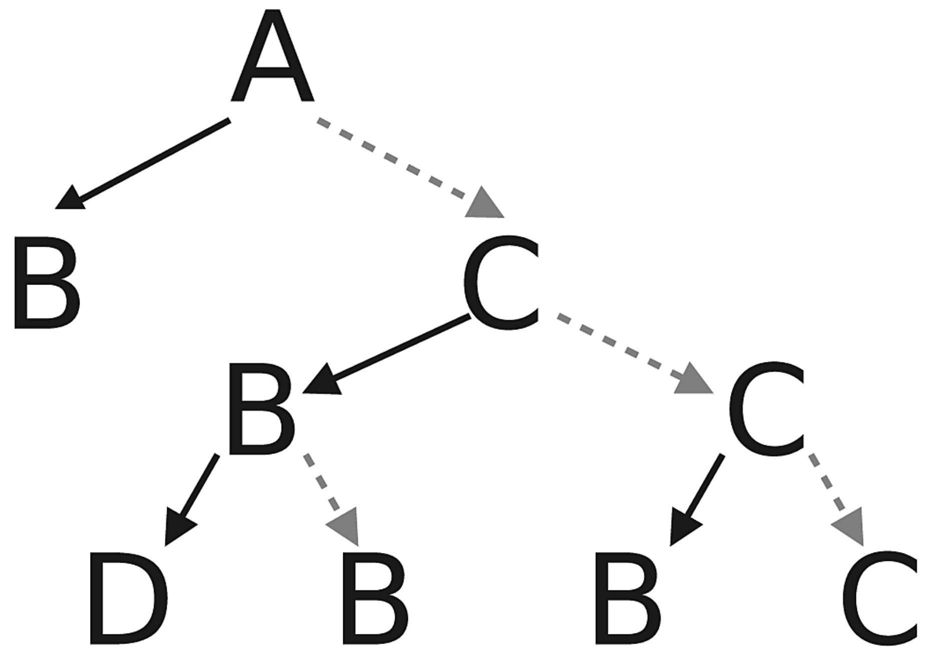
```
//===------------------------------------------------------------------===//
// This file is distributed under the MIT License.
// See LICENSE for details.
//
// The code is intentionally insecure so is only intended for testing purposes.
//
// Original copyright: Paul Kocher
//===------------------------------------------------------------------===//
/// \file
///
/// This is the sample function from the Spectre paper.
//===------------------------------------------------------------------===//
#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>

extern size_t array1_size, array2_size;
extern uint8_t temp, array2[], array1[];

void victim_function(size_t x) {
    if (x < array1_size) {
        temp &= array2[array1[x] * 512];
    }
}

int main(int argc, char **argv) {
    if (argc != 2) {
        printf("USAGE: %s <index>\n", argv[0]);
        exit(1);
    }

    int index = atoi(argv[1]);
    victim_function(index);
    printf("r = %d\n", temp);
    return 0;
}
~
~
~
~
~
~
~
~
"main.c" 36L, 1003C
```



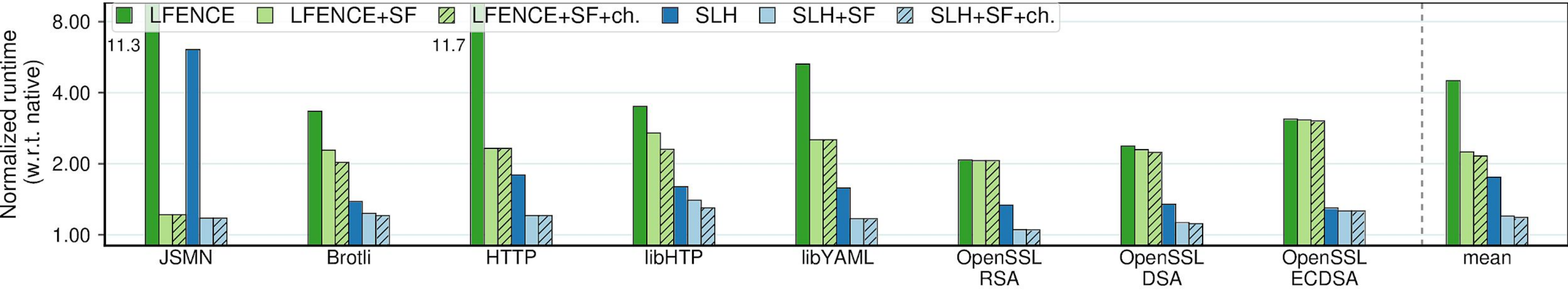
```
[SF], 1, 0xe850e3, 0x0, -8, 0xe85155
[SF], 1, 0xe8508d, 0x0, -8, 0xe85155
[SF], 1, 0xe850e3, 0x0, -8, 0xe85155
[SF], 1, 0xe8508d, 0x0, -8, 0xe85155
[SF], 1, 0xe850e3, 0x0, -8, 0xe85155
[SF], 1, 0xe8508d, 0x0, -8, 0xe85155
[SF], 1, 0xe850e3, 0x0, -8, 0xe85155
[SF], 1, 0xe8508d, 0x0, -8, 0xe85155
[SF], 1, 0xe850e3, 0x0, -8, 0xe85155
[SF], 11, 0xed7e9d, 0x0, 0, 0xed865e
[SF], 1, 0xeda182, 0x0, -8, 0xed9be9
[SF], 1, 0xeda1fc, 0x0, -8, 0xed9be9
[SF], 1, 0xeda26e, 0x0, -8, 0xed9be9
[SF], 1, 0xeda30c, 0x0, -8, 0xed9be9
[SF], 1, 0xeda383, 0x0, -8, 0xed9be9
[SF], 1, 0xeda3fd, 0x0, -8, 0xed9be9
[SF], 1, 0xeda495, 0x0, -8, 0xed9be9
[SF], 1, 0xed9cfe, 0x0, -8, 0xed9be9
[SF], 1, 0xed9eed, 0x0, -8, 0xed9be9
[SF], 1, 0xed9f5e, 0x0, -8, 0xed9be9
[SF], 1, 0xeda0e5, 0x0, -8, 0xed9be9
[SF], 1, 0xeda182, 0x0, -8, 0xed9be9
[SF], 1, 0xeda6da, 0x0, -8, 0xeda863
[SF], 1, 0xeda746, 0x0, -8, 0xeda863
[SF], 1, 0xeda7de, 0x0, -8, 0xeda863
[SF], 1, 0xeda66d, 0x0, -8, 0xeda863
[SF], 1, 0xeda6da, 0x0, -8, 0xeda863
[SF], 1, 0xeda746, 0x0, -8, 0xeda863
[SF], 1, 0xeda7de, 0x0, -8, 0xeda863
[SF], 1, 0xeda66d, 0x0, -8, 0xeda863
[SF], 1, 0xeda66d, 0x0, -8, 0xeda863
[SF], 1, 0xeda6da, 0x0, -8, 0xeda863
[SF], 1, 0xeda66d, 0x0, -8, 0xeda863
[SF], 1, 0xeda6da, 0x0, -8, 0xeda863
[SF], 1, 0xed93d2, 0x0, -8, 0xed9440
[SF], 1, 0xed9371, 0x0, -8, 0xed9440
[SF], 1, 0xed93d2, 0x0, -8, 0xed9440
[SF], 1, 0xed9371, 0x0, -8, 0xed9440
[SF], 1, 0xed93d2, 0x0, -8, 0xed9440
[SF], 1, 0xed9371, 0x0, -8, 0xed9440
[SF], 1, 0xed93d2, 0x0, -8, 0xed9440
[SF], 1, 0xed9371, 0x0, -8, 0xed9440
[SF], 1, 0xed93d2, 0x0, -8, 0xed9440#
{13:17}~/code/spectre/fuzzing/openssl ⇒ honggfuzz -N 10 -Q -n 1 -f ./fuzz/corpora/server -l openssl.log -- ./fuzz/
server ___FILE___ 2>&1 | analyzer.py collect -r openssl.log -o analyzed.json -b ./fuzz/server
```



(a) Control Flow Graph



(b) A's Simulation Tree


LFENCE
LFENCE+SF
LFENCE+SF+ch.
SLH
SLH+SF
SLH+SF+ch.
Normalized runtime (w.r.t. native)
8.00
4.00
2.00
1.00
11.3
11.7
JSMN
Brotli
HTTP
libHTP
libYAML
OpenSSL RSA
OpenSSL DSA
OpenSSL ECDSA
mean